5-2008

On Broadening Software Development Productivity Research to Serve Better Software Engineering Management

Doncho Petkov
Eastern Connecticut State University, petkovd@easternct.edu

Olga Petkova
Central Connecticut State University, petkovao@ccsu.edu

61F. On Broadening Software Development Productivity Research to Serve Better Software Engineering Management

Doncho Petkov
Eastern Connecticut State University,
petkovd@easternct.edu

Olga Petkova
Central Connecticut State University,
petkovao@ccsu.edu

Abstract
The unresolved problems of improving software engineering management require a broader systemic approach of investigating related issues like software development productivity. The paper links software engineering management to research on software cost estimation and on factors affecting software development productivity. It examines ways for the systemic incorporation of all issues influencing a software project through application of combination of methods from diverse paradigms.

Keywords
Software development management, Software cost estimation, Factors affecting software development productivity, Software development productivity.

1. Introduction
A better understanding of the factors affecting software development productivity and the relationships between them is essential for the improvement of the management of the information systems development process through improved cost and effort estimation (see Boehm, 2006). Boehm and Sullivan (1999) present thought provoking ideas on the need for renewal in the capabilities to reason about major software decisions in economic terms. They note, "The new software development techniques demand new estimation methods"(Boehm and Sullivan, 1999). One way to address this is the periodic calibration of existing cost estimations models, like COCOMOII to reflect the new development practices (e.g. see Boehm, Abts and Chulani, 2000). The introduction of agile approaches and internet-speed software development, poses new demands towards the improvement of our understanding about what drives software development (see Baskerville et al.(2003), Boehm (2006) and others).

Boehm and Sullivan (1999) point out that an important issue is to empower high-level managers to choose the best available economic reasoning techniques for use in their projects. The same authors conclude that "the pressures for improved cost-benefit and return on investment analyses are causing more software researchers and business analysis researchers to come together to integrate their knowledge and tools into more effective capabilities, not just for analysis, but also for more effective software management" (Boehm and Sullivan, 1999:945). This is another justification for the need of a deeper analysis of software development productivity.

The multiple dimensions of the factors affecting software development productivity represent a challenge for an interdisciplinary research to their nature. Past research on software development productivity and related areas such as cost and effort estimation was carried out predominantly within the paradigms of Computer Science and Software Engineering. Information Systems Development (ISD) is also the core of the field of Information Systems (see Hirschheim and

Klein, 2003). Hence, the field of software development productivity can be viewed as one of the areas of intersection of research interests within Information Systems, Software Engineering and Computer Science. More details on an analysis of the research in the three disciplines can be found in Glass, Ramesh and Vessey (2004). The contention of this paper is that software development productivity needs to be studied in an interdisciplinary manner even if we recognise differences in the core competencies of the disciplines. Very few authors have attempted to analyze software development productivity outside of the domain of competencies that is characteristic of software engineering and that was another motivation for this paper.

Hirschheim and Klein (2003) identify the four ISD process core competencies (organizational alignment of IT; user requirements construction, organizational implementation, and evaluation/assessment of IT artifacts) possessed by IS specialists and that distinguish them from Software Engineers (see SWEBOK, 2004). It is true that software engineering and to some extent computer science have changed over the years towards recognizing the role of the human element. Thus, Kemerer (1998) states that, "because of the significant role played by people, software engineering is already one of the computer science disciplines that is closest to the social sciences".

The importance of research on factors affecting software development productivity is both of a practical and theoretical nature. From the point of view of the needs of the software industry, there is a gap between the claims of publications in the area of software measurement regarding its usefulness and the actual degree to which such approaches are applied in practice (Pfleeger, 2000; Pfleeger et al., 2002). One of the ways to reduce this gap is to provide better methods for the assessment of factors affecting software development productivity that are intuitive and easy to use by managers and at the same time are rigorous from the point of the current state of decision making as a scientific field.

The majority of the research in software development productivity is only indirectly linked to it through the definition of productivity. It is primarily focused on effort and cost estimation or software project management and control. According to Boehm and Sullivan (1999:939), "…most of today's successful estimating procedures are largely empirical in nature. It would be good to have theoretical models that would capture, explain and permit us to reason from an understanding of the underlying dynamics…" This has been one of the inspirational ideas behind this research. Here, past work on factors affecting software development productivity is linked to the area of software process improvement and information systems development management in general, pointing to some work of interpretative nature that has been done recently in the Information Systems field. To the best of the authors' knowledge, this is the first attempt at a comprehensive analysis of the multifaceted work on the factors affecting software development productivity exceeding the domain of cost estimation and providing links to related publications within the broader goal of the improvement of software development management through methods enabling organizational learning.

The paper continues with a critical review of previous publications on issues related to software development productivity. It is followed by an analysis of research that is indirectly related to cost and effort estimation through improved understanding of what drives software development and how to manage better software development. At the end are provided some conclusions on potential future work.

2. Software development productivity and the factors affecting it

Since software is intangible, it is not possible to measure productivity directly (Sommerville, 2007). According to the same source, in software systems, what we really want to estimate is the cost of deriving a particular system with given functionality and that is only indirectly related to some tangible measures as those used by various methods from the literature.

The best definition of productivity as a process, according to Boehm (1987:44), is:

$$\text{Productivity} = \frac{\text{Outputs produced by the process}}{\text{Inputs consumed by the process}}$$

Thus, evaluation of productivity in a software project depends on the ways we define the inputs and the outputs of the software process. Following Boehm(1987), inputs to the software development process generally comprise labour, computers, supplies, and other support facilities and equipment. The same author points out that there might be other meaningful inputs in the organisational context of a particular environment and frequently one can use present-value monetary expressions as a uniform scale for various classes of resources as inputs.

Our concern in this paper will be with factors affecting productivity over the entire systems development life cycle. Some authors concentrate only on one phase or aspect of it. For example, historically, researchers were predominantly interested in programmer productivity.

2.1. Cost estimation models and factors affecting software development productivity

Empirical research into productivity in the system development process had its beginnings in the 1960s . It can be classified in various ways (see also Conte, Dunsmore and Shen (1986):

- field studies versus laboratory experiments;
- prescriptive or descriptive models;
- analytic models, e.g. COCOMO (Boehm, 1981), versus analogy-based models (Shepperd and Schofield, 1997);
- machine learning methods using neural networks (see Heisat, 2002) or Case Based Reasoning (CBR) (Mudhopadhyay et al, 1992).

Another classification reflecting the recent development in cost estimation involving a Bayesian analysis (see Boehm et al., 2000) and systems dynamics (see Abdel-Hamid and Madnick (1991); Madachy (1996)) is provided in Boehm and Sullivan (1999):

- expertise-based methods,
- model-based methods,
- regression-based methods,
- composite-Bayesian methods,
- learning-oriented methods and
- dynamics-based methods.

It can be noted, that this categorization does not mention analogy-based methods including Case Based Reasoning.

It has been argued by many researchers that the cost estimation models discussed in the literature do not seem to capture productivity factors very well. A major problem in most of the research surveyed has been the insufficient precision of the results. Thus, Kemayel et al.(1991:157) left 53.1% of software development productivity unexplained through their statistical analysis. Relatively little precision of different analytical models had been reported in Kemerer (1998), Jones (1991) and others. This is particularly true when a certain model is applied to an environment that is different from the one used as a basis for gathering statistical data for deriving the model Maxwell et al. (1999).

Pfleeger (2000) lists the best-reported values on parameters characterising the precision of ten well-known methods for effort and cost estimation and concludes that the statistics for most models are disappointing. This raises the question as to how successful the methods currently applied within the domain of software cost estimation are. A major concern limiting their usability and precision is the need to calibrate them according to the specific conditions of a given organisational environment. This need is partially recognised through the research on calibration of existing cost/effort estimation models with company specific data (see Maxwell et al. (1999), Jeffery et al. (2000)).

According to Subramanian and Breslawsli (1995), the poor performance of normative models, especially in foreign sites, has lead researchers to consider descriptive estimation techniques. These approaches estimate effort through the provision of a better insight into the estimation of the development effort. Such ideas are reflected in the work in the area of software effort estimation on analogy-based reasoning and case based reasoning (see Mudhopadhyay et al. , 1992; Shepperd and schofild, 1997; shepperd and Cartwright, 2001).

The issues related to the factors affecting software development productivity have been identified in the past as a major area of research in the literature on software productivity/cost/effort estimation (Maxwell et al., 1999), Bergeron and St-Arnaud, 1992). In most cases, these factors have been analysed with respect to the tuning or validation of a particular metric for cost/effort estimation. These factors are labelled differently in different models: in Boehm's (1981) COCOMO model, they are called cost drivers; Bailey and Basili (1981) name them as input to the model, while in Function Point Analysis (FPA) they are called Adjustment Factors (see Abran and Robillard, 1996).

Due to their large number, a certain classification of these factors is appropriate. Benbasat and Vessey (1980) discuss 19 factors grouped in seven classes: organizational operations characteristics, computer hardware characteristics, source language characteristics, programmer characteristics, programming problem characteristics. Kemayel et al. (1991) focus their attention on the controllable factors affecting software development productivity. According to them, controllable factors are those pertaining to the software process that a typical software manager has the latitude to determine. They investigate 33 controllable factors placed in three groups: factors pertaining to personnel; factors pertaining to the process; and factors pertaining to the user community. Finnie, Wittig and Petkov (1993) investigate 18 factors grouped in four groups: technical attributes, project attributes, developer attributes and user attributes.

It can be concluded that the major factors affecting software development productivity are those included in the contemporary version of COCOMO, known as COCOMOII model (see Boehm et al. , 2000) and in the Function Point Analysis approach (see Abran and Robillard,1996). The majority of the researchers in the field accept their formulations, although there are a number of

cases when additional factors are introduced in the literature. COCOMOII is synthesizing ideas from the original COCOMO and FPA. It uses Source Lines of Code and/or Function Points as the sizing parameter for inputs, adjusted for both reuse and breakage, a set of 17 multiplicative effort multipliers and a set of 5 exponential scale factors (Boehm et al., 2000).

A brief observation of the lists of the factors affecting software development productivity in various publications shows that a small number of them evolve with time and the development of technology. Thus, these days it seems inappropriate to consider the effect of memory size or storage devices, as did Benbasat and Vessey (1980). Another conclusion that comes to mind is the fact that various researchers focused their attention on quite different sets of factors, thus it is very difficult to compare results from previous surveys.

The diverse use of factors affecting software development in software engineering and in information systems justifies the need for their deeper analysis. One of the initial steps may involve some standardization of the definitions of certain factors used in well-established methods for cost and effort estimation like COCOMOII and Function Point Analysis for the whole field of software management. Thus, Rainer and Hall (2003) investigate the influence of 26 factors affecting software processes. These factors were identified through extensive computerised qualitative analyses of the text of past papers in the software engineering literature on software process improvement, which provides credibility to their list. At least six of them have some similarities with those factors featuring in COCOMOII, but their wording is different. One reason could be the fact that software productivity improvement (SPI) researchers are still seeking to identify the key factors that affect SPI programs (Rainer and Hall, 2003:19).

A relatively little researched issue is the importance of time in software productivity estimation. A well-known model of the error in estimates at various stages of a project is the Cone of Uncertainty in cost estimation, discussed in Boehm (2008). As far as factors are concerned, the later the estimate, the greater the percentage of the factors that are related to more accurate estimates (Benbasat and Vessey, 1980:253). They also claim that different factors could be used, depending upon the development phase at which the first global estimate of the development effort is made. Thus, estimators are encouraged to use specific methods and factors at specific phases. Similarly, Pfleeger et al. (2002) promote the need for early measurement in the software life cycle, but they also note that project managers are often intimidated by the effort required to track process measures throughout development.

A conclusion that can be drawn from an overall analysis of the various research attempts in the field of software development productivity is that there is no consensus on the relevant factors that have to be reflected in a particular method or model (see also Maxwell et al. (1996)). Most of the studies only include an analysis of a few factors, ignoring others. As most of the researchers indicate how complex the process of software development is, it can be assumed, that any attempts to drastically reduce the number of factors under concern may lead to over simplification of a model and its subsequent inadequate performance. In addition, there is usually little justification in the literature as to why certain factors are not included in a particular model. For example, user characteristics, application type and programming language are omitted from COCOMO.

Maxwell et al. (1999) note that in models such as COCOMO (see Boehm, 1981), the factors are treated as if they are independent, even though some are not. Furthermore, they conclude that some of the factors used in existing models for cost and effort estimation were not among the

underlying factors affecting productivity, hence the need for the development of a simple model based on the determination of a small number of independent factors (Maxwell et al.,1999). Determining the necessary factors is still an open research goal.

Therefore, there is a need to reduce the number of factors that may be included in a particular model to a reasonable quantity so that it may be sufficiently understandable and comprehensible. This problem requires further research into the nature of each factor, its relationships with other factors and their ranking according to their contribution to software development productivity. The latter issue shows the need for models guiding the prioritisation of software development productivity factors. The identification of a smaller number of factors that are most important within a particular project allows management efforts to be directed primarily towards those factors, thus leading to more effective corrective action if needed.

2.2. How objective can be software cost and effort estimates?

While qualitative methods were observed to be related to more accurate estimates, no clear conclusion could be drawn concerning the usefulness of quantitative methods (Bergeron and St-Arnaud, 1992). It is interesting to note that qualitative research based on psychometric measurements was generally neglected up until 1990, with few exceptions (e.g. see Hanson and Rosinski, 1985). Subsequently there has been a greater interest towards them (e.g. see Miranda, 2001; Shepperd and Cartwright, 2001).

A related point to the above question is the limitation of statistical variance theories in explaining social processes, including the process of systems development. As Robey (1994:443) concludes "...by conceiving of processes as systems of variables, the variance strategy affords little insight into the dynamics of the social processes it purports to explain. While some percentage of variance in one variable may be "explained" by the variation in another, little "explanation" of how and why social events occur is possible". According to him, in addition to valuable tools for research, such as the traditional science approaches, are also interpretive research methods that allow novel theoretical insights to be induced from both qualitative and quantitative data (Robey, 1994).

Research carried out by Maxwell et al. (1999) and others on the role of the company, country and the industrial/business environment is aiming at providing light on how the results from modelling the factors affecting software development can be generalised for different organizational environments. The issue of transportability of the results from one environment to another is an unresolved one, however. Many like Kemerer (1998), Conte et al (1986), and in particular Abdel-Hamid and Madnick (1991), have indicated the low utility of historical project statistics for cost and schedule estimation. Many of these criticisms are expressed as an insufficient capability of the models to reproduce their results in different environments. One major obstacle to the transportability of these models to environments that are different from those used for the original collection of data to build the model appears to be a lack of understanding of the factors explaining the differences in productivity among projects (Maxwell et al, 1999:787).

Software cost or effort estimation involves well-established analytical methods. However, their application is a time consuming activity, which aims also to be objective. That is in contradiction with the role of subjectivity in the estimation process. Thus selecting particular adjustment factors always requires an expert opinion by the estimator. We may observe that all of the

"algorithmic" methods or models actually involved an element of subjectivity in choosing the productivity factors, e.g. in choosing the cost driver ratings in the COCOMO model. An excellent review of expert estimation in software development effort can be found in Jorgensen (2004). However, the majority of the theory and practice of software measurement community shows that there is a deep concern within researchers about the involvement of subjectivity in this process. To answer whether or not subjectivity has some role to play in the assessment of factors affecting software development productivity requires further investigation.

Subjectivity is related also to the issue on how possible is to reproduce effort estimates on similar projects in different environments. The ability to reproduce results is a significant feature of the scientific approach (see Ackoff, 1973; Checkland, 1999). It cannot be met in this area because of the human factors involved. On the other hand, some researchers like Kemerer (1998), correctly note that the socio-cultural background and the computing tradition in a particular country inevitably affect results in cost and effort measurement. All the above probably indicates a need to rethink the traditional claims to strive towards pure objectivity in cost and effort estimation and to accept that it is not achievable in real IS project management, due to the human factor involved. These ideas are further explored in the next section.

3. Research on factors affecting software development productivity dealing with broader issues of software management

According to Boehm and Fairley (2000:24), besides "setting budgets and schedules and supporting make-or-buy analyses, software estimation techniques have several additional decision support uses:

- supporting negotiations or trade-off analyses among software cost, schedule, quality, performance and functionality;
- providing the cost portion of a cost-benefit or return on investment analysis;
- supporting software cost and schedule risk analyses and risk management decisions;
- and supporting software quality and productivity improvement investment decisions".

The factors affecting software development productivity play an important role in the management of the Information Systems development process. They influence the way in which the work of software development teams is organised and controlled (see Abdel-Hamid (1991), Boehm (2008) and others). An important contribution to understanding team and individual issues in software development is the work of Watts Humphrey (see Humphrey, 1996). The factors play an important role in the broader field of software measurement (see Pfleeger et al., 2002).Most of the above topics represent a separate stream of research in the Information Systems and Software Engineering literature with specific epistemology.

3.1. Early attempts for analyzing the factors affecting software development productivity and learning about software project dynamics issues

Most of the IS research reported in the literature is limited to statistical analyses of cause-effect type of relationships between two or three factors only. An alternative way for modelling the relationships between factors affecting software development productivity is the application of a multi-criteria decision analysis approach, the Analytic Hierarchy Process in Finnie et al. (1993). The latter paper suggests a simple model that allows the prioritization of the factors according to their importance for a given project. The smaller number of factors that are identified as

important factors for a particular development situation allows management to concentrate only on these.). Support for the idea of this investigation can be found in Rainer and Hall (2003:15).

Finnie et al (1993) note also that although past research seems to assume independence of factors affecting software processes, attention should be directed at how such factors relate to each other.
Their modelling approach can be extended further in the direction of capturing all the interrelationships between the factors affecting productivity using an extension of AHP for models with feedback (Saaty, 1996). Such models are also known as Analytic Network Process (Saaty, 1996). Such a model of the factors affecting software development productivity is explored in Petkova (1999). In essence, it serves a somewhat similar purpose as the use of Bayesian Belief Networks in modelling software projects (see Stamelos et al., 2003) and the application of Systems Dynamics for a similar purpose, which is discussed next.

Abdel-Hamid and Madnick (1991) summarize their work conducted in the 1980s on modeling the software development micro-world through systems dynamics. Subsequent accounts can be found in Madachy(1996) and Madachy (2007). This approach explores the relationships between various factors affecting software development focusing on the existing feedback influences between them. It acknowledges that software development, a dynamic and complex process, requires new ways of thinking in order to improve the current software environment. Systems dynamics involves thinking in circles and considering interdependencies, closed loop causality versus straight-line thinking, seeing the system as a cause rather than an effect, internal versus external orientation, thinking dynamically rather than statically, operational versus correlation orientation (Madachy, 1996). Systems dynamics modeling can provide insights by investigating virtually any aspect of the software process at a macro or micro level.

Besides systems dynamics, other techniques for modelling software process dynamics have been used as well. According to Ramil and Lehman (1999) these include Petri nets, fuzzy logic, and combinations of fuzzy logic and systems dynamics. Bayesian belief networks were applied as another way to support expert judgment in software cost estimation (see Stamelos et al., 2003). One possible direction for further research is to compare the expressive power of models applying the above approaches, including Systems Dynamics and the Analytic Network Process. The major issue of concern in such a comparative study should be the determination of the effort associated with the application of a particular method. The latter is important since according to Ramil and Lehman (1999), “modelling of software process dynamics has, however, not attracted widespread interest. The level of expertise, data and resources required to build and calibrate such models did not appear justified in a perception of limited benefit”. Similar criticism can be stated about another direction line of research in the 1980s and 1990s by Scacchi, aimed at development of knowledge-based systems that model software production (see Scacchi, 1995). Hence, the need for simpler modelling approaches that still capture the complexity of software development and the interrelationships between the factors affecting it.

3.2. Towards broader interpretive holistic approaches for understanding software development management perceived as a learning process

Kitchenham et. al.(2002) point out the need for software cost estimation approaches and models that incorporate the instinctual ways that humans organize and manipulate the information at their disposal. Software development is treated as a learning process for the client and the service provider recently also by Champion et al. (2006). Such learning is affected by many sources of

change in a software project, that bring the need to consider uncertainty and emergent requirements in software projects as discussed by Boehm (2008). According to him, the best way to reduce the uncertainty in software development is to buy information to avoid risk. Boehm (2008) lists for that purpose as possible techniques prototyping, simulating, benchmarking, market trend analysis, analyzing relative costs and benefits. We would like to add that these methods for reducing uncertainty could be applied within a broader framework supporting organizational learning. The factors supporting organizational learning in software development are investigated in Van Solingen et al. (2000). Systems dynamics modelling is an early example of a method supporting organizational learning in the context of a software process (Madachy, 1996) but it cannot capture well all subjective issues in a project. Hence, we propose the idea to link the fields of software measurement and organisational learning in efforts towards improvement of the ISD process and show how it can be implemented with methods that go beyond systems dynamics.

Another approach supporting organizational learning for improved understanding of factors affecting software development productivity within a specific software project is presented in Petkova and Roode (1999). It draws on the growing number of attempts in IS research towards the application of qualitative approaches and methodological pluralism(see Hirschheim and Klein, 2003). Based on the analysis of the research in software metrics, we can conclude that an investigation of factors affecting software development productivity is a complex problem, with many intertwined sub-problems, and as such, it can be classified as a "messy" problem, to use the term coined first by Ackoff (1973). It is an important and worthwhile problem, both from a theoretical and practical point of view. A framework is proposed in Petkova(1999) and Petkova and Roode (1999) for its analysis that is relatively easy to use, which allows for the incorporation of all relevant factors affecting software development productivity in a holistic way. It incorporates available quantitative and qualitative data on a particular project environment, as well as the expertise of those involved in its management. It combines techniques from soft systems thinking and multi-criteria decision analysis, enabling experiential learning about the relationships between the factors involved. This approach is in line with the call by Scacchi (1995) for development of setting-specific models of software production, which "can be tuned to better account for the mutual influence of product, process and setting characteristics specific to a process".

Petkova and Roode (1999) implemented a pluralist systemic framework for the evaluation of the factors affecting software development productivity within a particular organizational environment. It combines techniques from several paradigms; stakeholder identification and analysis, from SSM (Checkland, 1999) and the Analytic Hierarchy Process (Saaty, 1996). The framework is based on the principles of Multimethodology, a powerful systemic metatheory for organizing an intervention in Operations Research (see Mingers, 2001) through mixing methods from different methodologies. This framework is complementary to the existing cost estimation approaches and does not aim to replace them. The result of its application is a better expert judgment in the usage of other existing methods for software cost and effort estimation, as well as for IS project management.

There are very few accounts of the use of Soft Systems Methodology (SSM) (see Checkland, 1999) in the mainstream SE literature. More details on that and a recent example of combining SSM and UML for business process modelling are presented in Sewchurran and Petkov (2007). It is significant that Boehm has recognised the potential of SSM as he quotes Checkland in a recent paper: "… software people were recognizing that their sequential, reductionist processes

were not conducive to producing user-satisfactory software, and were developing alternative SE processes (evolutionary, spiral, agile) involving more and more systems engineering activities. Concurrently, systems engineering people were coming to similar conclusions about their sequential, reductionist processes, and developing alternative 'soft systems engineering' processes, emphasizing the continuous learning aspects of developing successful user-intensive systems" (Boehm, 2006a).

The challenges of software and systems engineering for improvement of software development productivity may be addressed better through the application of a systems approach. This is advocated recently by Petkov et al. (2008). The latter work extends some of the ideas in Abdel-Hamid and Madnick (1991), Madachy (1996), Petkova and Roode (1999). Most of the suggestions on integrating IS, SE and systems thinking in Petkov et al. (2008) relate to issues of organizational learning facilitated by a systems approach. Soft systems methods have a significant track record of producing results in management (se Checkland, 1999) and Mingers, 2001) and accordingly their potential in software development management is stressed in Petkov et al. (2008).

According to Boehm (2006a), "recent process guidelines and standards such as the Capability Maturity Model Integration (CMMI), ISO/IEC 12207 for software engineering, and ISO/IEC 15288 for systems engineering emphasize the need to integrate systems and software engineering processes". The same author further proposes a new process framework for integrating software and systems engineering for 21st century systems, and improving the contractual acquisition processes.

Boehm (2006b) presents a deep analysis of the history of SE and of the trends that have emerged recently. These include the agile development methods (see Baskerville et al. (2003)]); commercial off-the-shelf software and model driven development. The traditional software development world, characterized by software engineering advocates uses plan-driven methods, which rely heavily on explicit documented knowledge. Plan-driven methods employ project planning documentation to provide broad-spectrum communications and rely on documented process plans and product plans to coordinate everyone (Boehm, 2006b). The late 1990s saw something of a backlash against what was seen as the over-rigidity contained within plan-driven models and culminated in the arrival of agile methodologies, which rely heavily on communication through tacit, interpersonal knowledge for their success.

The above paragraphs indicate major developments in the area of software process management and software development management. Research in this direction should not ignore previous related work in software cost and effort estimation and software development management. An example of how past work is integrated with the new ideas in software management is the topic on what happens at the level of large systems-of-systems, discussed in Lane and Boehm (2007), dealing with some issues related to cost and effort estimation for large and complex software projects).

4. Conclusion

This paper attempted to review the past research on factors affecting software development productivity. The links between this issue and the broader aspects of software development management were also explored. Our findings show that:

- the models of relationships between such factors have so far not contributed significantly to the improvement of the precision of estimates for development effort or costs;
- the number of factors affecting software development productivity quoted in the literature exceeds seventy. There are no universal classifications accepted in the fields of Information Systems, Software Engineering and Computer Science. There is not enough evidence about justifying the selection or exclusion in a particular model of certain factors;
- there is a need to provide a link between work on software measurement with research on the management of the software development process;
- the stakeholders in a project in some cases are easy to identify, but more often, there are subtle stakeholders that need to be uncovered to avoid a potential over- simplification of the understanding of the role of a project and its environment;
- a better understanding of the relationships between the factors affecting software development productivity can be achieved through application of methods promoting organizational learning. The result of this is an improved Information Systems Development process;
- there is a need for relatively simple approaches to identify which are the most appropriate factors affecting software development productivity for a given organisation and project, how they relate to each other, and how their ranking evolves at different stages of the software development process.

Different organizational environments may justify the inclusion of various factors affecting software development productivity. It seems that the choice of those that need to be considered in a given situation must be more flexible. Such decisions depend on the actual dynamic circumstances of a particular IS project and how well these are understood. The survey showed that very few researchers attempted to address this issue in its full complexity.

Another dimension of possible future analysis can be concerning the benefits from mixing methods from different paradigms and methodologies in frameworks promoting better understanding of the factors affecting software development productivity compared to the use of a single methodology like systems dynamics. A starting point could be raising the awareness of a particular research community of what another one is doing. The results of the review point that a pluralist approach, breaking down the paradigmatic isolation between various research methodologies may lead to improvement of our understanding to handle the complexity of improving software development productivity.

There have been a number of comparative studies of various cost and effort estimation methods (see Kemerer (1998); Kitchenham et al. (2002) and others). On the other hand, there have been few attempts to apply approaches promoting better Information Systems Development process management through organizational learning as the several attempts discussed in the paper. To the best of our knowledge, there have been no comparative studies of such approaches promoting organizational learning within a software development management environment.

With the exception of Boehm, most of the authors working in software cost and effort estimation focus on their own research agenda with little comparison of how it fits within the greater body of knowledge on software development management. Most of that research is from the perspective of Software Engineering with few exceptions like the IS oriented work reported by Scacchi (1995) and Petkova and Roode (1999). Hence, there is a need for integrative research in a systemic way the problem of software development management and the related productivity

factors across the current disciplinary divide in order to produce better practical, relevant and rigorous understanding of what drives software development productivity.

References

Abdel-Hamid, T.K. and Madnick, S., Software Process Dynamics, Prentice Hall, Englewood Cliffs, NJ., 1991

Abran, A. and Robillard, P.N. , Function Point Analysis: An Empirical Study of Its Measurement Process, IEEE Transactions on Software Engineering, 22(12), 895-910, 1996

Ackoff, R., Science in the Systems Age: Beyond IE, OR and MS, Operations Research, 21, 661-669,1973.

Bailey, J.W. And Basili, V.R., A Meta-Model for Software Development Resource Expenditures, Proceedings of Fifth International Conference on Software Engineering, 107-116,1981.

Baskerville, R., Ramesh, B., Levine L., Pries-Heje, J., and Slaughter, S., Is Internet-Speed Software Development Different, IEEE Software, November/ December, 70-77, 2003.

Benbasat, I. and Vessey, I., Programmer and Analyst Time/Cost Estimation, MIS Quarterly, 4(2), 31-43,1980.

Bergeron, F. and St-Arnaud, J.,SOS: Estimation of Information Systems Development Efforts: A Pilot Study, Information and Management, 22, 239-254,1992.

Boehm, B.W., Software Engineering Economics, Prentice-Hall, Englewood Cliffs, NJ.,1981.

Boehm, B.W., Improving Software Productivity, IEEE Computer, 20(9), 42-57,1987.

Boehm, B., Some future trends and implications for systems and software engineering processes. Systems Engineering, 9(1), 1-19, 2006a.

Boehm, B., A view of 20th and 21st century software engineering, Proceeding of the 28th international conference on Software Engineering. Shanghai, China, 12 – 29, 2006b.

Boehm, B., Making a difference in the software century, *IEEE Computer*, 41(3),32-38,2008.

Boehm, B. and Sullivan, K.,Software Economics: Status and Prospects, Information and Software Technology, 41, 937-946, 1999.

Boehm, B. and Fairley, R., Software Estimation Perspectives, IEEE Software, November/December, 22-26, 2000.

Boehm, B, Abts, C. and Chulani, S., Software development cost estimation approaches - A survey. Annals of Software Engineering, 10: 177-205, (2000)

Champion, D., Stowell, F., & O'Callaghan, A. (2005). Client-Led Information System Creation (CLIC): navigating the gap. *Information Systems Journal*, (15), 213-231.

Checkland, P., Systems thinking, systems practice. Wiley, Chichester, 1999.

Conte, S.D., Dunsmore, H.E. And Shen,V.Y., Software Engineering Metrics and Models, The Benjamin Cummings Publishing Co. Menlo Park, 1986.

Finnie, G.R., Wittig, G.E. and Petkov, D.I., Prioritizing Software Development Productivity Factors Using the Analytic Hierarchy Process, Journal of Systems and Software, 22(2), 129-139, 1993.

Glass, R., Ramesh, V & Vessey, I. (2004). An analysis of research in computing disciplines. *Communications of ACM*, 47(6), 89-94.

Hanson, S.J. and Rosinski, R.R., Programmer Perceptions Of Productivity And Programming Tools, Communications Of The ACM, 28(2), 180-189, 1985.

Heisat, A., Comparison Of Artificial Neural Network And Regression Models For Estimating Software Development Effort, Information And Software Technology, 44, 911-922,2002.

Hirschheim, R and Klein H.K., Crisis In The IS Field? A Critical Reflection On The State of the IS Discipline, Journal of the Association for Information Systems, 4 (5), 237-293, 2003.

Humphrey, Watts S, Using A Defined and Measured Personal Software Process. *IEEE Software 13*(3): 77-88,1996.

Jeffery, R., Ruhe, M. and Wieczorek, I., A comparative study of two software development cost modeling techniques using multi-organizational and company specific data, Information and Software Technology, 42, 1009-1016, 2000.

Jones, C., Applied Software Measurement: Assuring Productivity and Quality, McGraw Hill, New York., 1991.

Jørgensen, M., A Review of Studies on Expert Estimation of Software Development Effort, J. Systems and Software, 70 (1-2), 37–60, 2004.

Kemayel, L., Mili, A. and Ouderni, I., Controllable Factors for Programmer Productivity: A Statistical Study, Journal of Systems and Software, 16(2), 151-163, 1991.

Kemerer, C.F.,Progress, Obstacles, and Opportunities in Software Engineering Economics, Communications of ACM, 41(8), 63-66, 1998 .

Kitchenham, B., Pfleeger, S.L., Mccoll, B. and Eagan, S., An empirical study of maintenance and development estimation, The Journal of Systems and Software, 64 (1), 57-77, 2002.

Lane, J.A. and Boehm, B., System-of-systems cost estimation: analysis of lead system integrator engineering activities. Information Resources Management Journal, 20(2), 23-32, 2007.

Madachy, R.J., Process Modeling with Systems Dynamics, Proceedings of the Eight Software Engineering Process Group Conference, Atlantic City NJ, May., 1996.

Madachy, R. *Software Process Dynamics*, Wiley, 2007.

Mingers, J., Multimethodology- mixing and matching methods. in Rosenhead J. and Mingers J. (Eds), Rational analysis for a problematic world revisited, Wiley, Chichester, 2001.

Maxwell, K.D., Wassenhove, L.V. and Dutta, S., Software Development Productivity of European Space, Military, and Industrial Applications, IEEE Transactions on Software Engineering, 22(10), 706-718, 1996.

Maxwell, K., Van Wassenhove, L. and Dutta, S., Performance Evaluation of General and Company Specific Models in Software Development Effort Estimation, Management Science, 45 (6), 787-803, 1999.

Miranda, E., Improving subjective estimates using paired comparisons, IEEE Software, January/February, 87-91, 2001.

Mudhopadhyay, T., Vicinanza, S.S. And Prietula, M.M., Examining the Feasibility of a Case-Based Reasoning Model for Software Effort Estimation, MIS Quarterly, June, 155-169, 1992.

Petkova, O., A framework for evaluation of factors affecting software development productivity, PhD thesis, University of Pretoria, South Africa, 1999.

Petkova, O. and Roode, J. D., An application of a framework for evaluation of the factors affecting software development productivity in the context of a particular organizational environment, South African Computer Journal, 24, 26-32, 1999.

Petkov, D., Edgar-Nevill, D, Madachy R., O'connor, R., Information systems, software engineering and systems thinking – challenges and opportunities, in print: Journal on Information Technology and the Systems Approach, 1 (1), 2008.

Pfleeger, S.L., Software Engineering: Theory and Practice, Prentice Hall Inc., 2000.

Pfleeger, S.L., Jeffery, R., Curtis, B., And Kitchenham, B., Status Report On Software Measurement, IEEE Software, March-April, 33-43, 2002.

Rainer, A. and Hall, T., A quantitative and qualitative analysis of factors affecting software processes, The Journal of Systems and Software, 66, 7-21, 2003.

Ramil, J.F. and Lehman, M.M., Modeling Process Dynamics in Software Evolution Processes – Some Issues, submitted to the Workshop on Software Change and Evolution (SCE'99), Los Angeles, May, 1999. Available at http://www.doc.ic.ac.uk/~mml/feast2/papers/pdf/619.pdf

Robey, D.,Modeling Interpersonal Process during System Development: Further Thoughts and Suggestions, Information Systems Research, 5(4), 439-445, 1994.

Saaty, T., The Analytic Network Process, RWS Publications, Pittsburgh, 1996.

Scacchi W., Understanding Software Productivity, In D Hurley (Ed). Advances in Software Engineering and Knowledge Engineering, Vol 4, 33-70, 1995.

Sewchurran, K. and Petkov, D. (2007). A systemic framework for business process modeling combining soft systems methodology and UML. Information Resources Management Journal, 20(3), 46-62.

Shepperd, M. and Schofield, C., Estimating Software Project Effort Using Analogies, IEEE Transactions on Software Engineering, 23(12), 736-743, 1997.

Shepperd, M. and Cartwright, M., Predicting with sparse data, IEEE Transactions on software engineering, November, 27 (11), 987-998, 2001.

Sommerville, I., Software Engineering, 8th Ed., Pearson, Harlow. 2007.

Stamelos, I., Angelis, L., Dimou, P., and Sakellaris, E., On the use of Bayesian belief networks for the prediction of software productivity, Information and Software Technology, 45 (1), 51-60, 2003.

Subramanian, G.H. and Breslawski, S., An Empirical Analysis of Software Effort Estimate Alterations, Journal of Systems and Software, 31,135-141, 1995.

Van Solingen, R., Berghout, E., Kusters, R., and Trienekens, J., From process improvement to people improvement: enabling learning in software development, Information and Software Technology, 42, 965-971, 2000